Solutions. Experts. Insights.

SEI TECHNOLOGIES FORUM

Software Engineering Institute | Carnegie Mellon

Agile Development and Software Architecture: Understanding Scale and Risk

Ipek Ozkaya

Research, Technology and Systems Solutions (RTSS) Program

Ozkaya is a senior member of the SEI technical staff within the Architecture-Centric Engineering (ACE) Initiative in the Research, Technology, and System Solutions (RTSS) Program. Her current interests and projects are in developing empirical methods for improving software development efficiency and system evolution with a focus on software architecture practices, software economics, and requirements management.

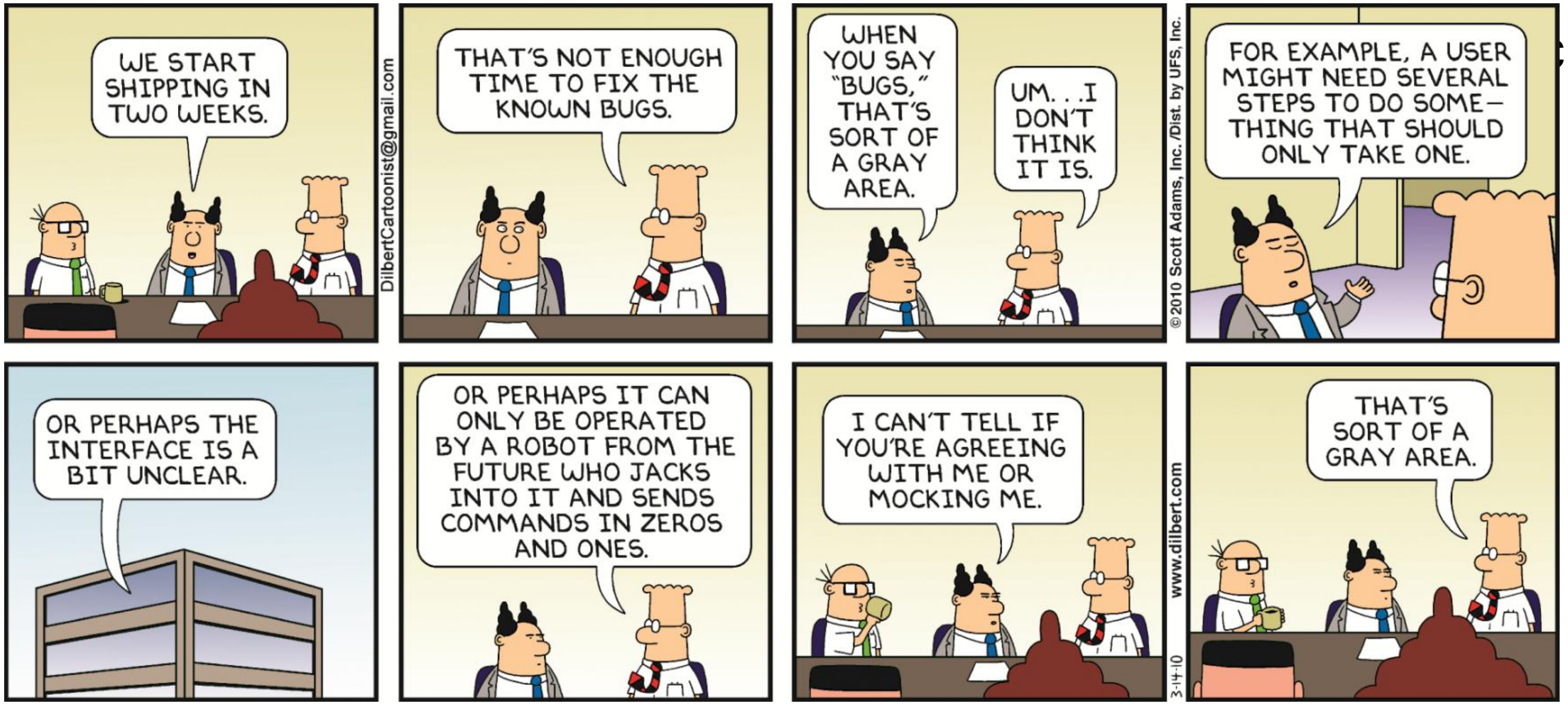
.

Report Documentation Page		*Form Approved OMB No. 0704-0188*
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.		
1. REPORT DATE **24 OCT 2011**	2. REPORT TYPE	3. DATES COVERED **00-00-2011 to 00-00-2011**
4. TITLE AND SUBTITLE **Agile Development and Software Architecture: Understanding Scale and Risk**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University ,Software Engineering Institute,Pittsburgh,PA,15213**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR’S ACRONYM(S)
		11. SPONSOR/MONITOR’S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**38**	

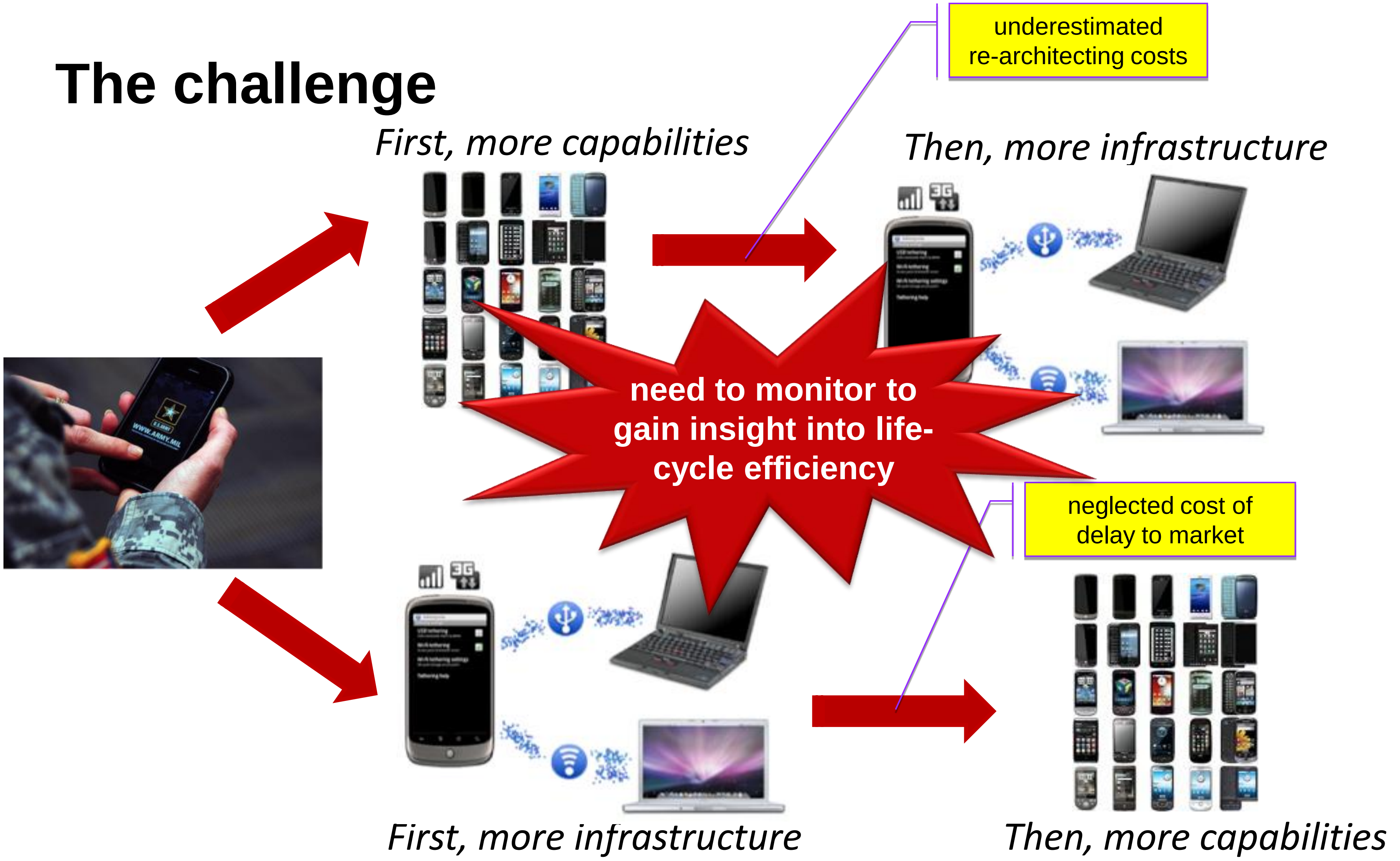
Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

The challenge


WE START SHIPPING IN TWO WEEKS.
THAT'S NOT ENOUGH TIME TO FIX THE KNOWN BUGS.
WHEN YOU SAY "BUGS," THAT'S SORT OF A GRAY AREA.
UM. . .I DON'T THINK IT IS.
FOR EXAMPLE, A USER MIGHT NEED SEVERAL STEPS TO DO SOMETHING THAT SHOULD ONLY TAKE ONE.
OR PERHAPS THE INTERFACE IS A BIT UNCLEAR.
OR PERHAPS IT CAN ONLY BE OPERATED BY A ROBOT FROM THE FUTURE WHO JACKS INTO IT AND SENDS COMMANDS IN ZEROS AND ONES.
I CAN'T TELL IF YOU'RE AGREEING WITH ME OR MOCKING ME.
THAT'S SORT OF A GRAY AREA.
DilbertCartoonist@gmail.com
©2010 Scott Adams, Inc./Dist. by UFS, Inc.
www.dilbert.com
3-14-10

Tradeoffs and their dependencies must be supported by both Agile software development and architecture practices

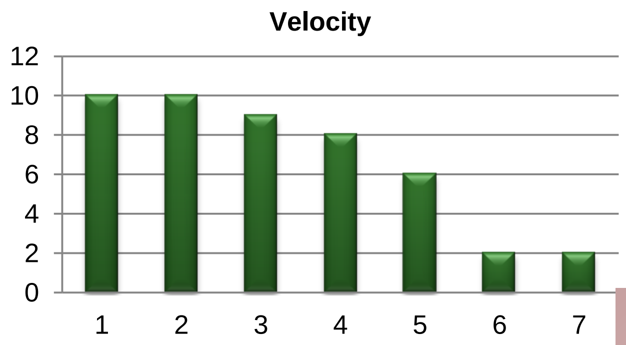
The challenge


First, more capabilities
underestimated re-architecting costs
Then, more infrastructure
need to monitor to gain insight into life-cycle efficiency
neglected cost of delay to market
First, more infrastructure
Then, more capabilities

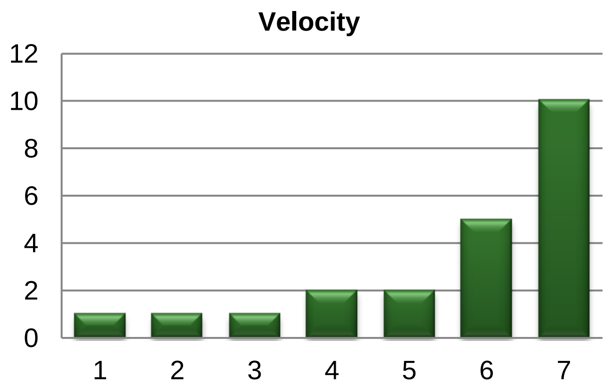
Brown, N., Nord, R., and Ozkaya, I. "Enabling Agility Through Architecture." *Crosstalk 23,* 6 (Nov./Dec. 2010): 12–17.

Increased visibility into delivery

Focus on Priority

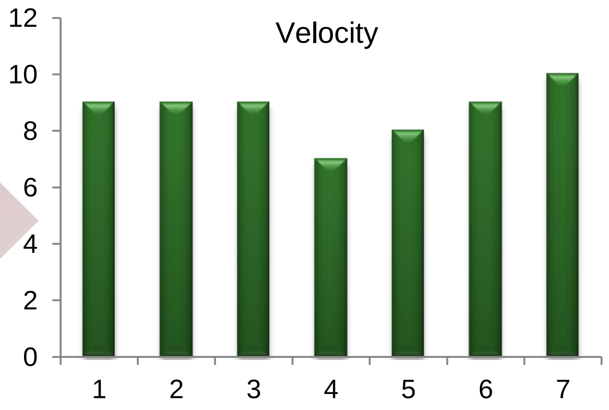


Focus on Cost



Use metrics to monitor & select development tasks

Focus on Integrated Value



Agenda

Symptoms of failure

Concepts of scale and root-cause analysis

Tactics that can help

- Align feature and system decomposition.
- Create an architectural runway.
- Use matrix teams and architecture.

Symptoms of failure

- Teams (e.g., Scrum teams, product development teams, component teams, feature teams) spend almost all of their time fixing defects, and new capability development is continuously slipping.

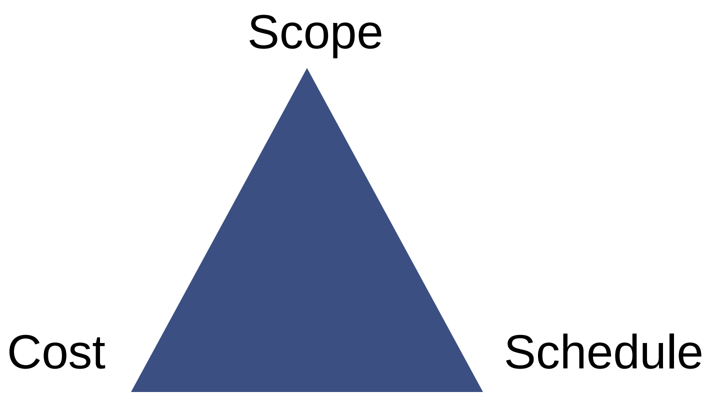
- Integration of products built by different teams reveals that incompatibility defects cause many failure conditions and lead to significant out-of-cycle rework in addition to end-to-end fault-tolerance failure.

- Progress toward meeting milestones is unsatisfactory.

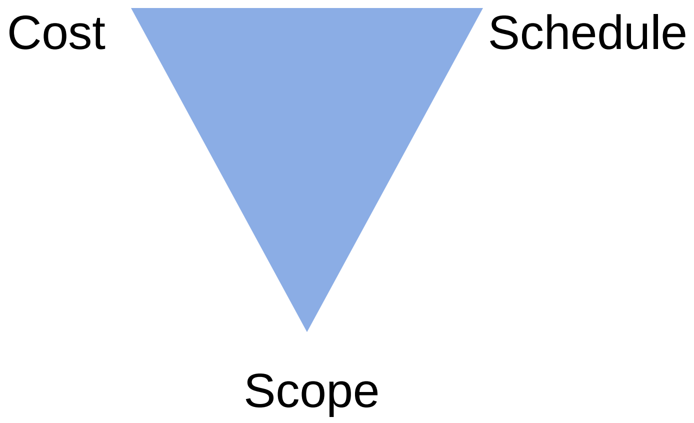
Scope drivers

Fundamental project management concerns are essential to keep in mind:

- If the *schedule* needs to be shorter, you may see an increase in *cost* and a decrease in *scope.*
- If *cost* becomes an issue, you may see a decrease in *scope* or an increase in *schedule.*
- If *scope* is increased, you may see an increase in both *cost* and *schedule.*

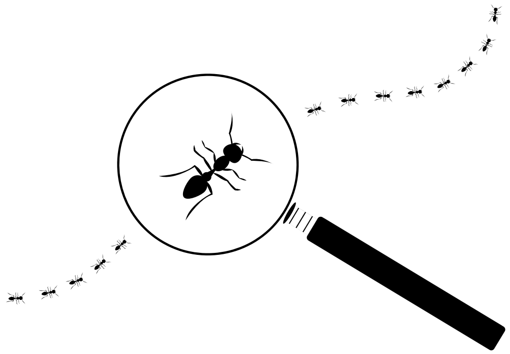


Traditional approach : Fixed scope driving cost and schedule



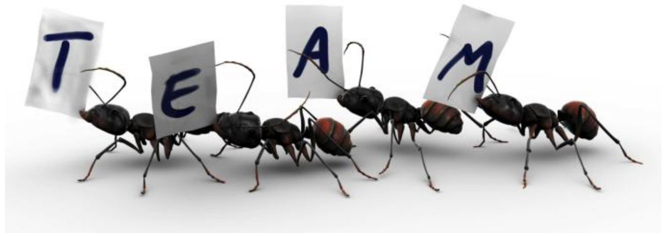
Agile project management approach: Fixed cost and schedule driving scope

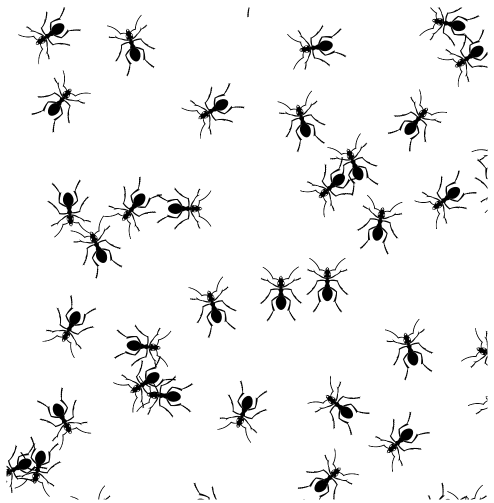
A closer look at scale: Scope



- Is the project in a new domain or technology?
- Does the project have new requirements such as standards compliance, system testing, and integration lab environments, or does it simply have more features, elements, and relationships?
- Is there a need to align systems engineering and software development activities?

A closer look at scale: Team





- Are there multiple teams that need to interact, both internal and external to the organization?

- What are the dependencies between the work products of system and software engineers?

- Have you considered the end-to-end success of features that may require resources from multiple teams?

A closer look at scale: Time

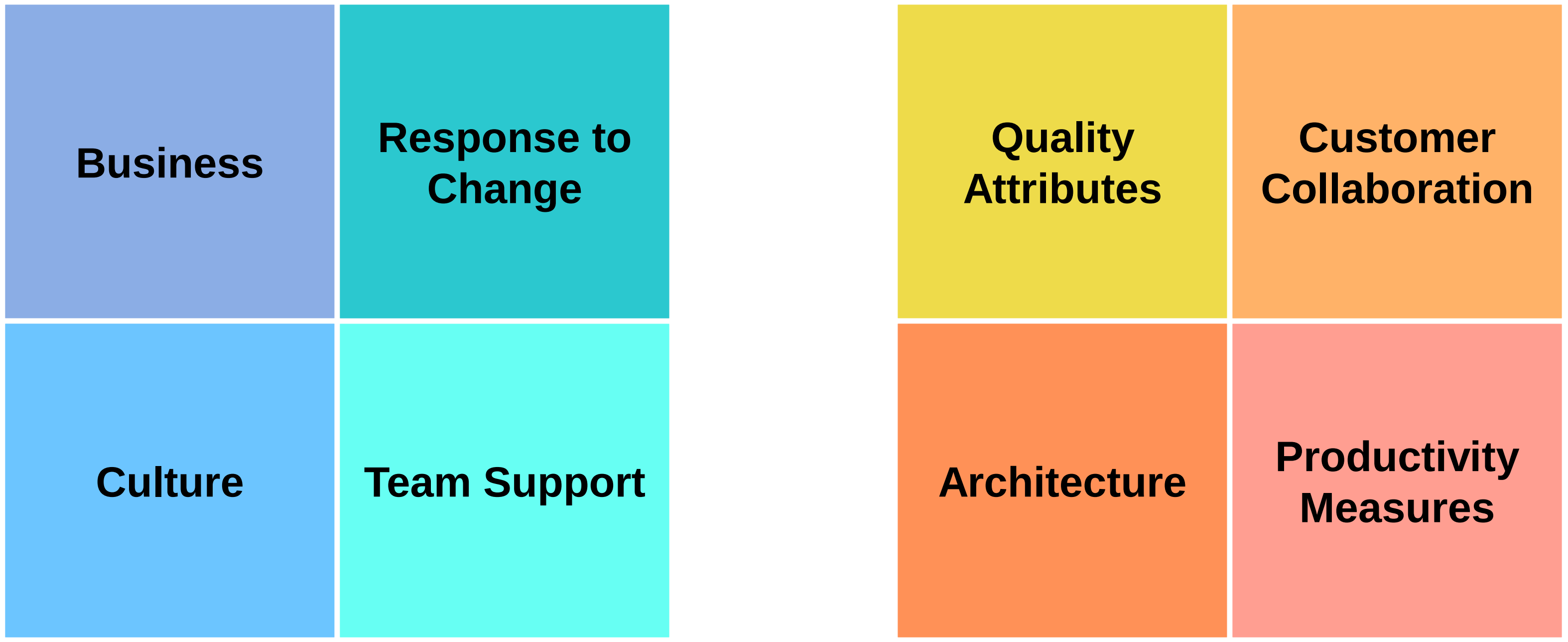
- Does the work require different schedule constraints for releases?
- How long is the work product expected to be in service?
- How important are sustainability and evolution?

Polling question

Are you currently doing development in a large-scale context that can be captured by extended scope, team size, or timelines of scale?

1. Large team size
2. Larger than normal scope
3. Longer development roadmap
4. Product expected to be in service for a long time
5. At least two of the above

Root-cause analysis

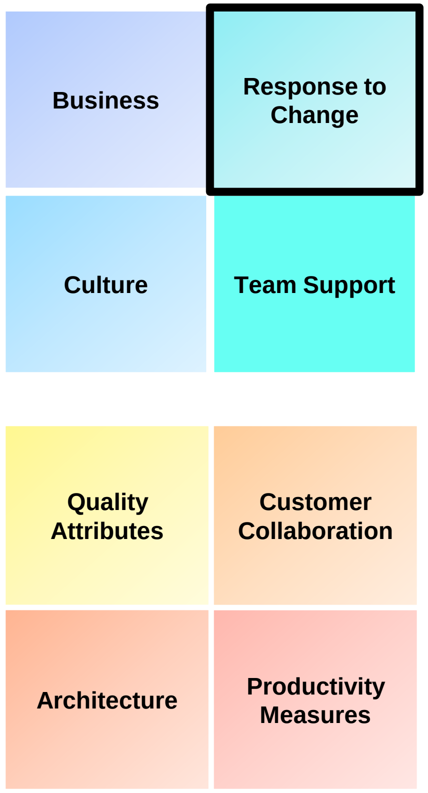


Investigate both technical and nontechnical areas, looking at both Agile software development and software architecture fundamentals.

Root-cause analysis

Response to change

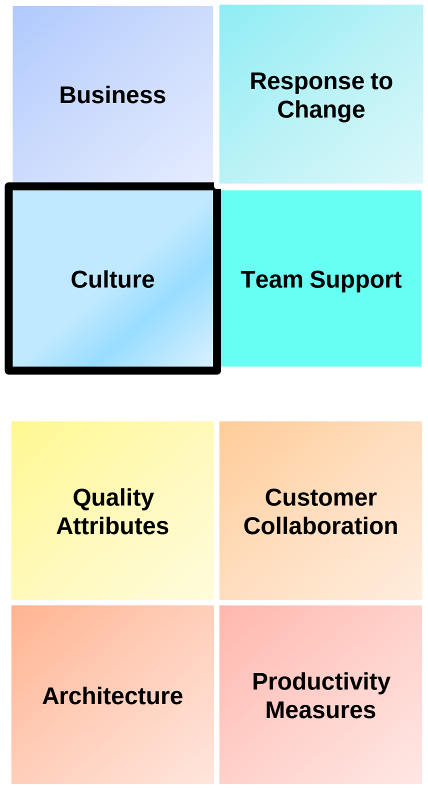
- Dynamic environment and changing requirements are understood.
- Necessary technology and processes are identified to respond to change.
- Impact of uncertainty on the project is acknowledged.
- Waste is identified and tradeoffs managed (e.g., technical debt and defects).



Root-cause analysis

Culture

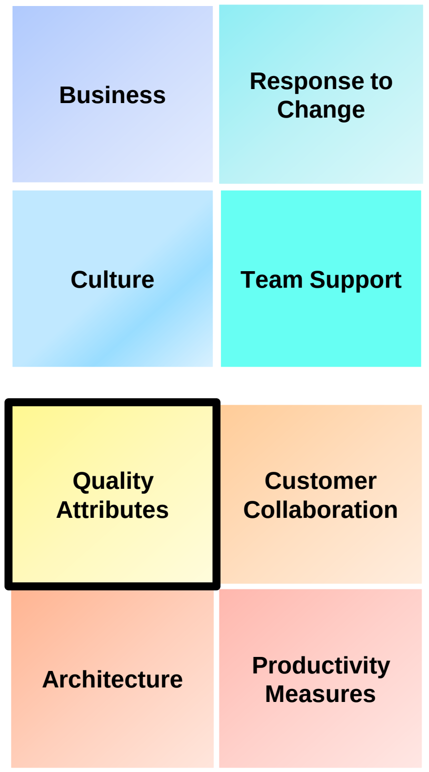
- People are made available (internal and external), including an appropriate number of people who have the right skills and knowledge and clear responsibilities.
- Team members are motivated and empowered by many degrees of freedom.
- Clear communication among teams and team members is established.
- There is high-level management support.



Root-cause analysis

Quality attributes

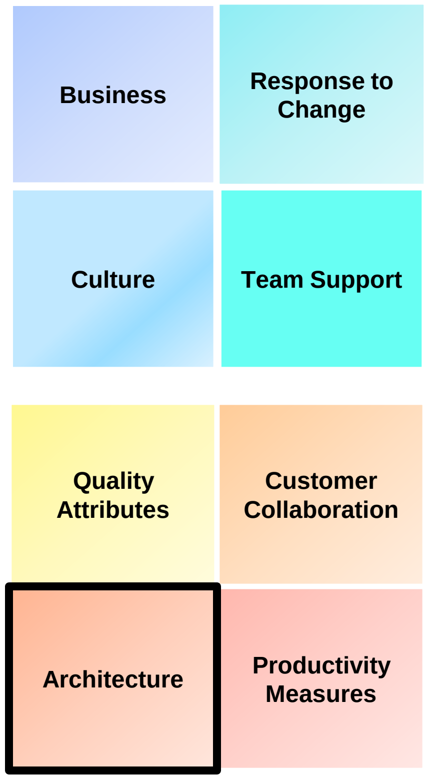
- The importance of quality attribute requirements is understood.
- Quality attribute requirements are defined and tied to business goals.
- Means for analysis of necessary quality attributes are in place and used to predict system properties.
- Measurement environment is in place to monitor the implemented system quality and "done" criteria.



Root-cause analysis

Architecture

- Evidence is provided that the architecture satisfies quality attribute requirements.
- Appropriate functional requirements are assigned to architecture elements.
- Architectural issues (e.g., technical debt) are tracked and managed.
- Timeline of critical architectural decisions is clear and scheduled.



Tactics to consider

Align feature and system decomposition.

Create an architectural runway.

Use matrix teams and architecture.

Align feature and system decomposition

Dependencies between stories & supporting architectural elements	Understanding the dependencies between stories and architectural elements enables staged implementation of technical infrastructure in support of achieving stakeholder value.
Dependencies among architectural elements	Low-dependency architectures are a critical enabler for scaling up Agile development.[1]
Dependencies among stories	High-value stories may require the implementation of lower value stories as precursors.[2]

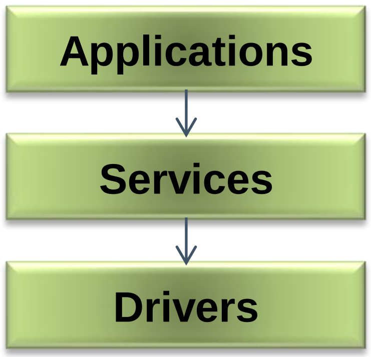
1. Poppendieck, M., and Poppendieck, T. *Leading Lean Software Development.* Addison-Wesley Professional, 2009.

2. Denne, M., and Cleland-Huang, J. *Software by Numbers*. Prentice Hall, 2003.

Align feature and system decomposition

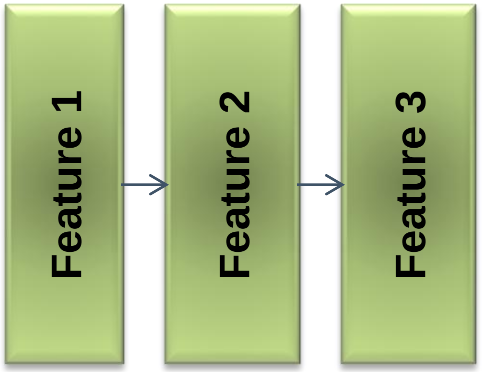
Tension between high-priority features (vertical decomposition) versus common reusable services (horizontal decomposition)

Infrastructure-driven approach



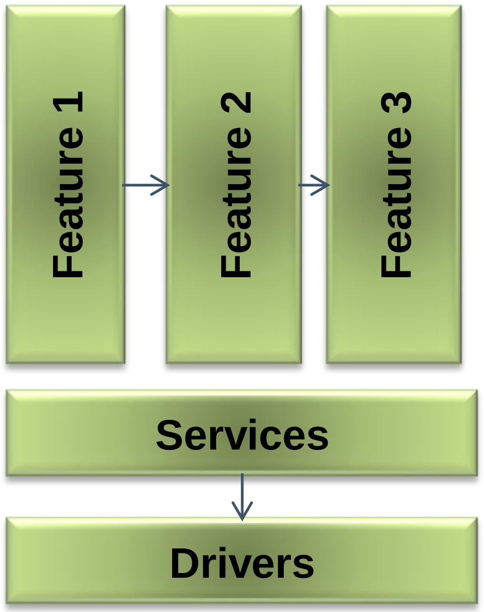
Horizontal decomposition (e.g., layers)

Feature-driven approach

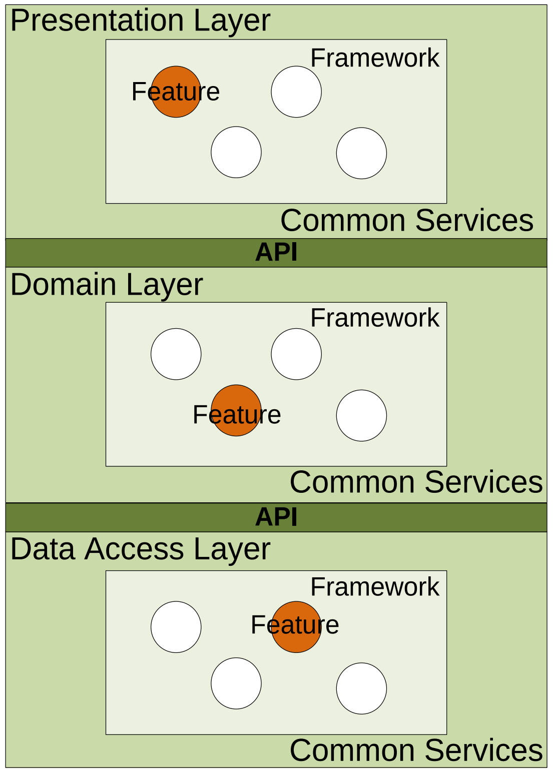


Vertical decomposition (e.g., subsystems, features)

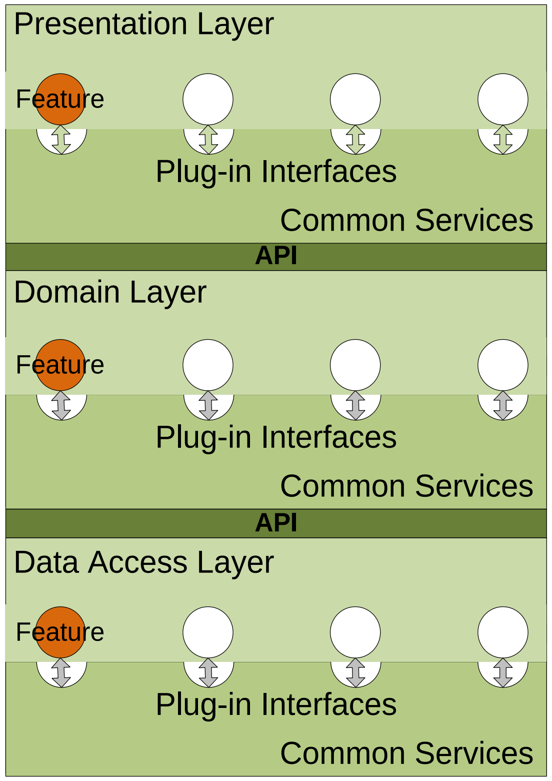
Hybrid approach



Align feature and system decomposition
Two examples



Layered architecture with frameworks



Layered architecture with plug-ins

Decouple teams and architecture to ensure parallel progress as the number of teams increases.

○ Unimplemented feature
● Feature

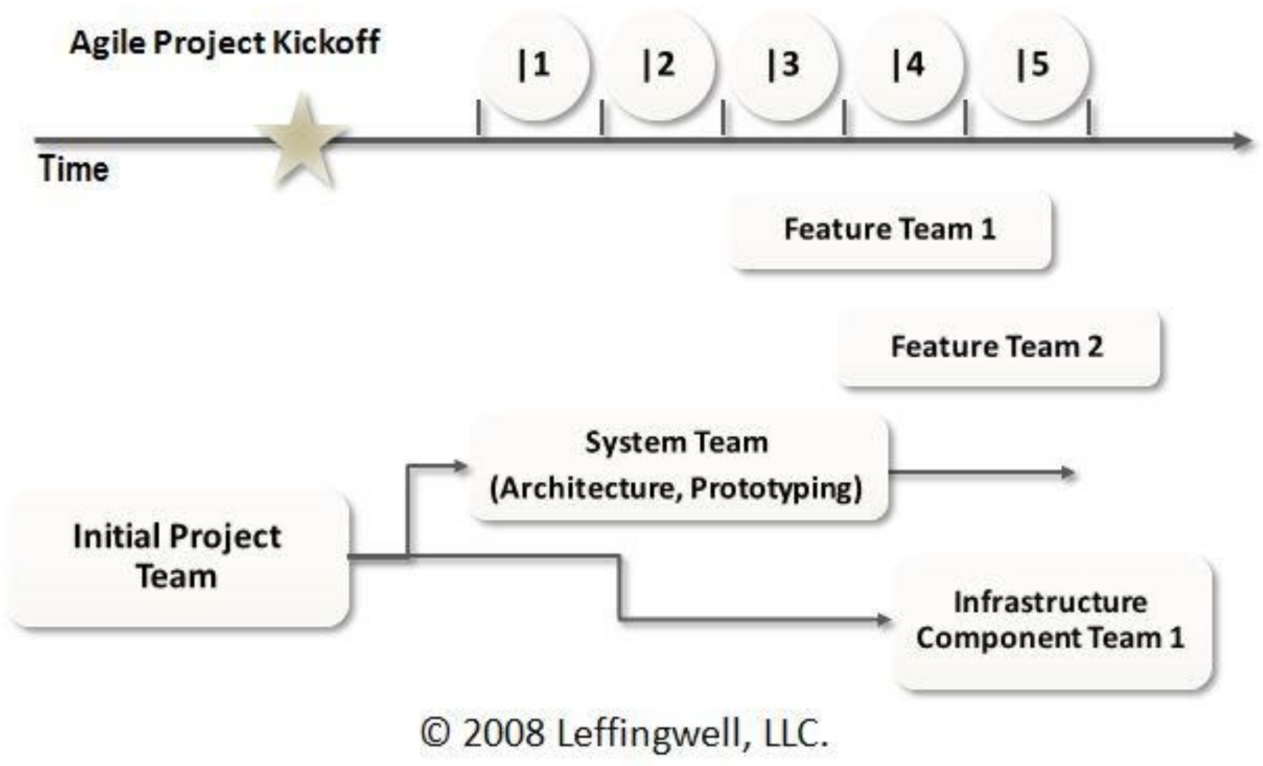
Create an architectural runway

The architectural runway provides the degree of architectural stability to support the next *n* iterations of development.

In a Scrum project environment, the architectural runway may be established during Sprint 0.

- Sprint 0 might have a longer duration than the rest of the sprints.

Create an architectural runway

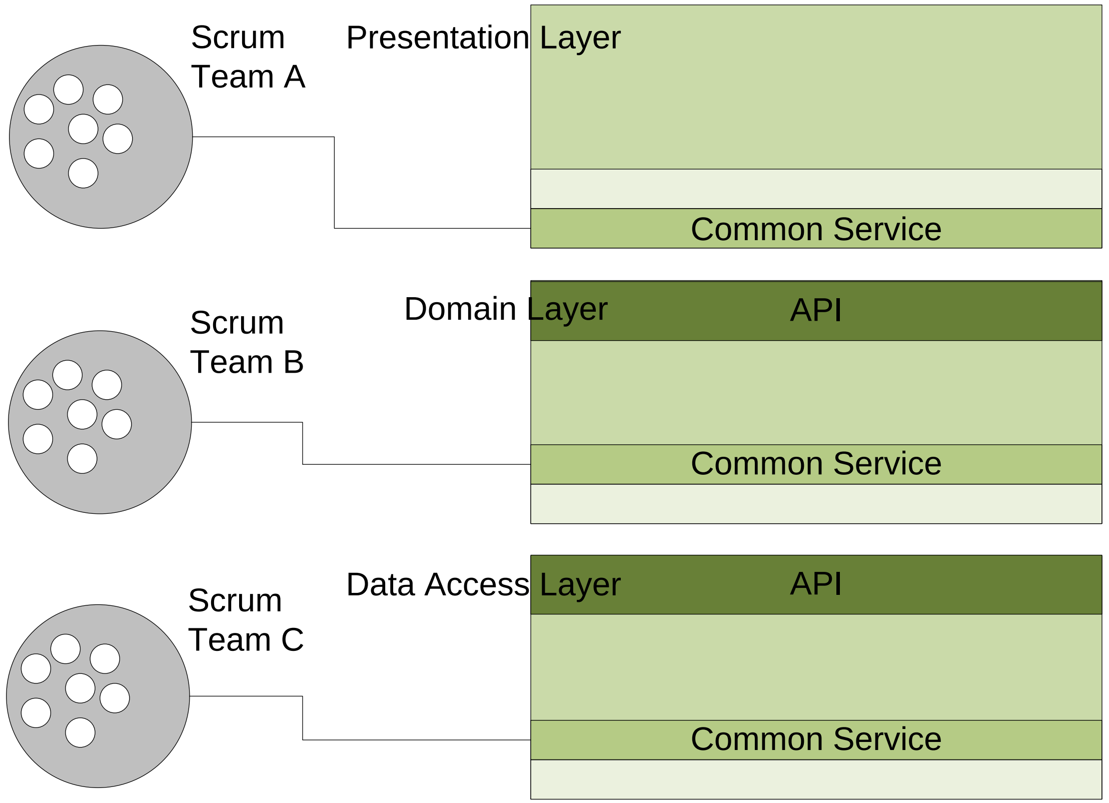


The bigger the system, the longer the runway.
Leffingwell, Martens, Zamora

Leffingwell, D. *Scaling Software Agility*. Addison-Wesley, 2007.
http://scalingsoftwareagility.wordpress.com/2008/09/09/enterprise-agility-the-big-picture-10-the-system-team

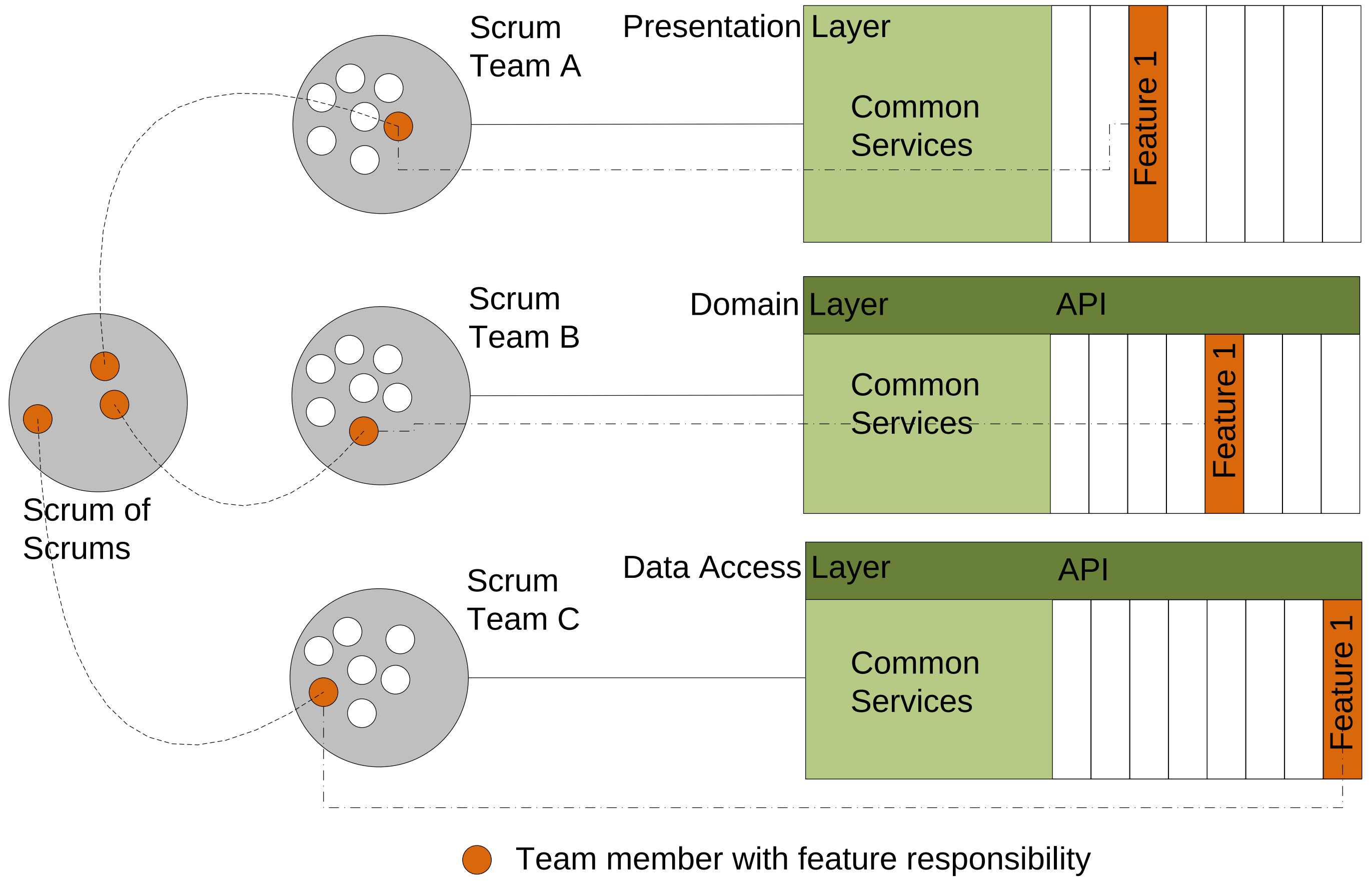
Use matrix teams and architecture

Establishing the infrastructure



Use matrix teams and architecture

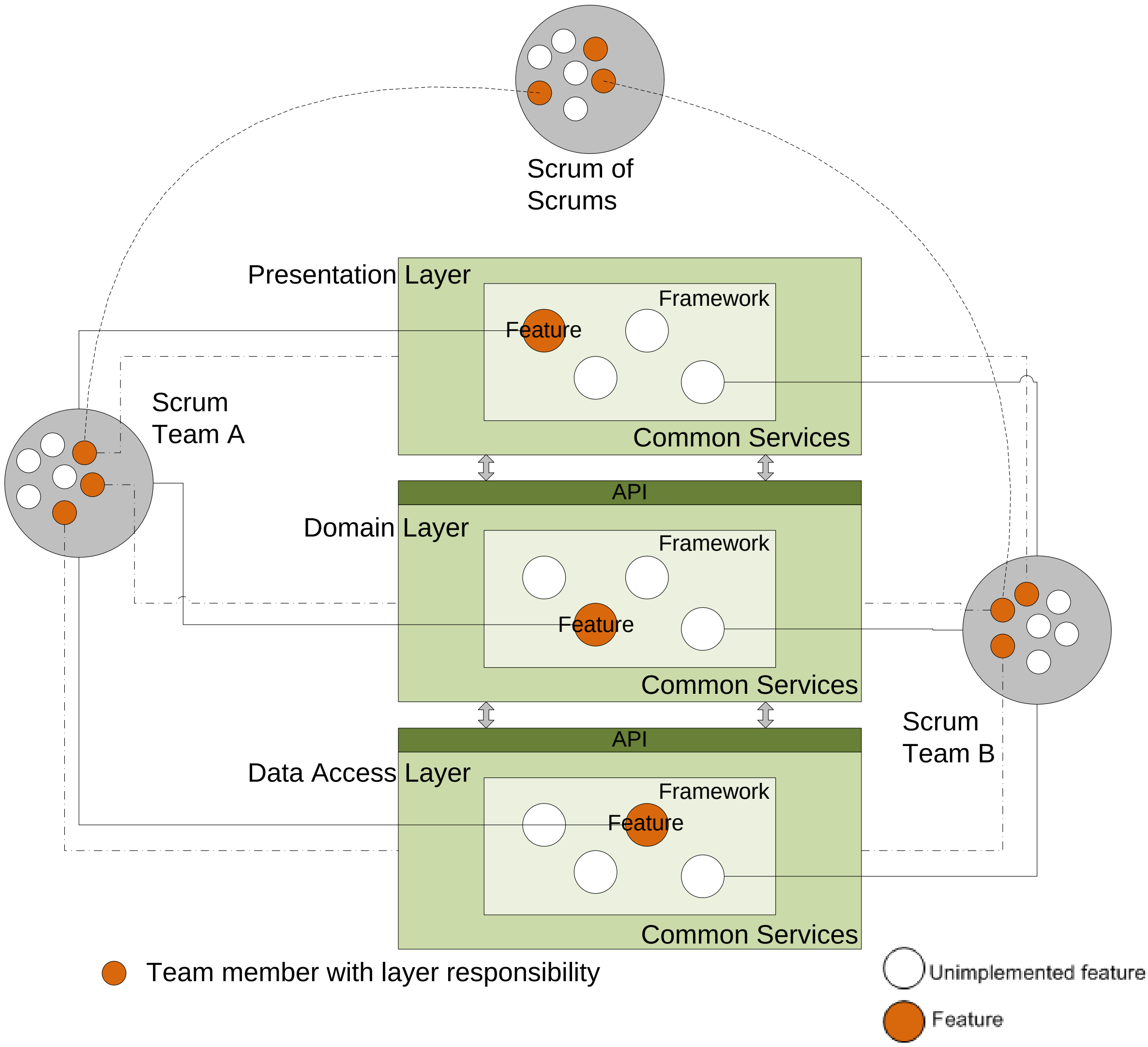
Feature development in parallel



● Team member with feature responsibility

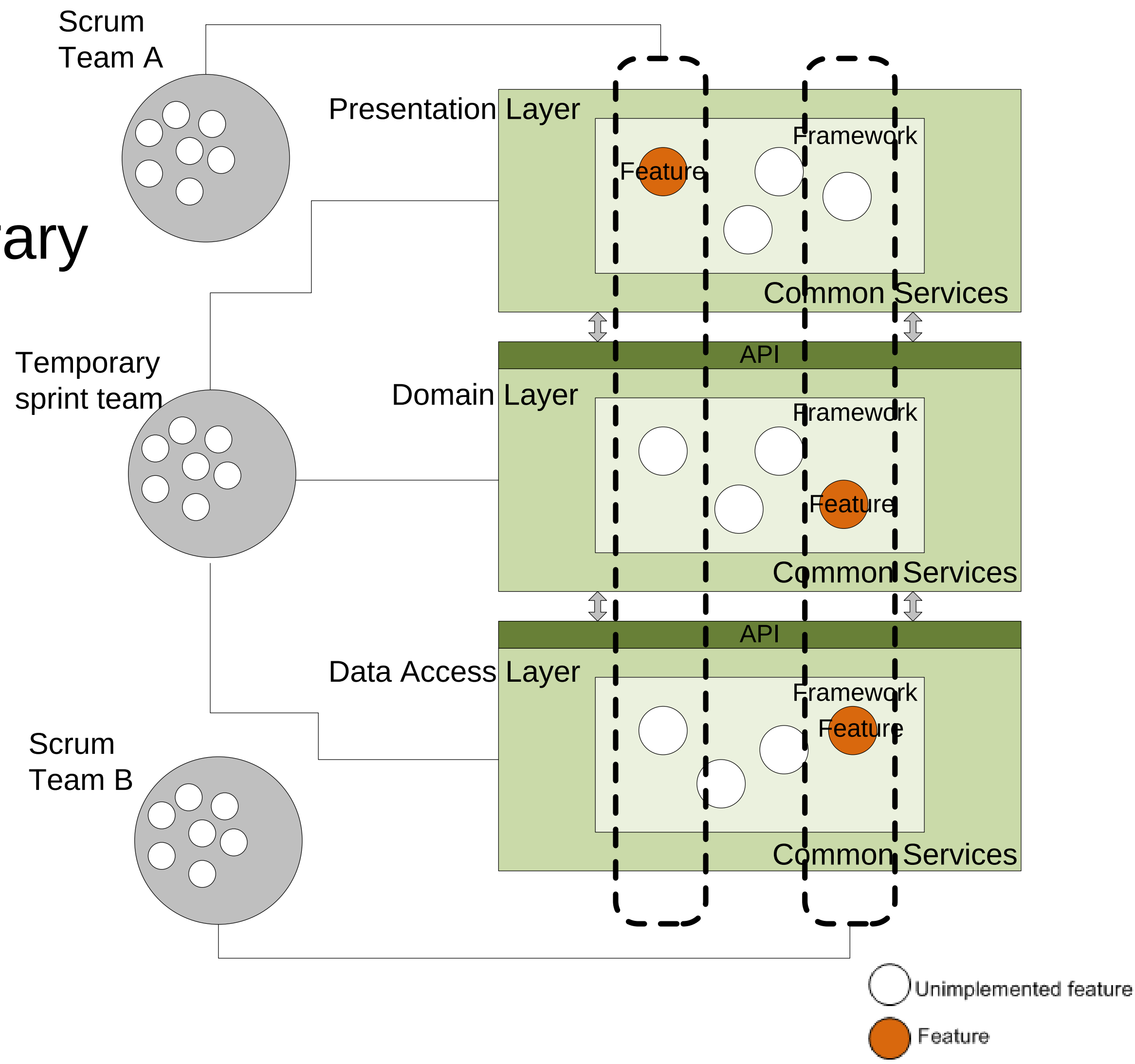
Use matrix teams and architecture

Different teams are assigned to different features, and some team members are assigned to keep layers and framework consistent.



Use matrix teams and architecture

Different teams are assigned to different features, and a temporary team is assigned to prepare layers and frameworks for future feature teams.



Root-cause analysis: Typical problem 1

Symptom

- Scrum teams spend almost all of their time fixing defects, and new feature development is continuously slipping.

Root-cause
<u>Inability to manage scope and time at scale</u>

- Initial focus was "general" rather than "product specific."
  - Time pressure to deliver became the top priority.
  - The team delivered an immature product.
  - A plethora of variation parameters interact detrimentally.
- There are three different cycles:
  - Customer release (annually, many variants); IV&V Testing (quarterly, 4 variants), and Developmental (monthly, 1 variant)

Solution

Stabilize the architecture.

- Build an architecture for current products.
  - Rules, guidelines
  - Over a few time boxes
- Reduce the number of “variant parameterizations.”
- Make everyone play from the same sheet music.
- Postpone adding new features.

Replan the release cycles/time boxes.

Revisit the testing strategy/team assignments against variants.

Root-cause analysis: Typical problem 2

Symptom

- Integration of products built by different Scrum teams reveals that incompatibility defects cause many failure conditions and lead to significant out-of-cycle rework.

Root -cause
<u>Inability to manage teams at scale</u>

- Cross-team coordination is poor, even though there are many coordination points and much time spent.
- Different teams have different interpretations of interfaces.
- The product owner on each Scrum team does not see the big picture.
- A mismatch exists between the architecture and Scrum development.

Solution

Stabilize to remove failures.

- Postpone adding new features.

Identify and collapse common services across teams.

Use an architectural runway.

- A system that has an architectural runway contains existing or planned infrastructure sufficient to allow incorporation of current and near-term anticipated requirements without excessive refactoring.
- An architectural runway is represented by *infrastructure* initiatives that have the same level of importance as the larger scale requirements epics that drive the company's vision forward.

Root-cause analysis: Typical problem 3

Symptom

- Progress toward meeting milestones is unsatisfactory.

Root-cause
<u>Inability to manage teams and scope at scale</u>

- Mapping of features to software components per Scrum cycle is disorganized.
- Some new features are unused in each cycle, causing wasted effort.
- Developer assignment to teams is inflexible.

Solution

Build more architectural views to align features between teams.

Reorganize teams to better fit iteration and release workloads.

Create matrix teams to clean up unused features.

Final thoughts

No one tactic alone can take any project to success.

Systematic root-cause analysis is essential for understanding risks arising in large-scale software development.

There are different aspects of scale that may need to be managed with different approaches, such as scope, team, and time.

Embracing the principles of both Agile software development and software architecture provide improved visibility of project status and better tactics for risk management.

- Align feature and system decomposition.
- Create an architectural runway.
- Use matrix teams and architecture.

References

Ambler, S. *The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments*. IBM developerWorks, 2009. https://www.ibm.com/developerworks/mydeveloperworks/blogs/ambler/entry/agile_scaling_model?lang=en

Brown, N., Nord, R., and Ozkaya, I. “Enabling Agility Through Architecture.” *Crosstalk 23,* 6 (Nov./Dec. 2010): 12–17.

Denne, M., and Cleland-Huang, J. *Software by Numbers*, Prentice Hall, 2003.

Kruchten, P. “What Color Is Your Backlog?” Agile Vancouver talk, 2009. http://files.me.com/philippe.kruchten/vuldw4

Larman, C., and Voddle, B. *Scaling Lean & Agile Development.* Addison-Wesley, 2009.

Leffingwell, D. *Scaling Software Agility*. Addison-Wesley, 2007. http://scalingsoftwareagility.wordpress.com/2008/09/09/enterprise-agility-the-big-picture-10-the-system-team

Poppendieck, M., and Poppendieck, T. *Leading Lean Software Development*. Addison-Wesley Professional, 2009.

Upcoming

Forthcoming SEI Technical Report on

Managing Agility at Scale:

A Software Architecture Perspective

Bachmann, F., Nord, R., Ozkaya, I., Wojcik, R., Wood, W., and Brown, N.

IEEE Software Special Issue on Technical Debt

Guest Editors: Philippe Kruchten, Robert Nord, and Ipek Ozkaya

http://www.computer.org/portal/web/computingnow/swcfp6

Contact Information

Ipek Ozkaya
Research, Technology, and System Solutions Program
Architecture Practices Initiative
Email: ozkaya@sei.cmu.edu

Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

As projects continue to grow in scale and complexity, effective collaboration across geographical, cultural, and technical boundaries is increasingly prevalent and essential to system success. SATURN 2012 will explore the theme of “Architecture: Catalyst for Collaboration.”